Java Performance

Session 407

Java Performance

Jim Laskey and Victor Hernandez
Java Runtime Technologies

Goal of This Session

- Understand performance of your Java application
- Introduce performance improvements to HotSpot 1.4 JIT Compiler
- Introduce performance opportunities with Java 2 Standard Edition v1.4 APIs

Presentation Overview

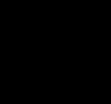
- HotSpot Compilation
- New API in Java 2 Standard Edition v1.4
- Summary
- Q&A

HotSpot Compilation

Jim Laskey
Technical Lead, Java VM Compilers

HotSpot 1.4 Compilation Technologies

- Class Hierarchy Analysis (CHA)
- Dynamic compilation
- Full-speed debugging
- Aggressive inlining
- Intrinsics
- LIR—Low-level Intermediate Representation



CHA—Class Hierarchy Analysis

- Scenario
  - 80% of classes are never subclassed (leaf/final)
  - Few overshadowed methods
  - instanceof does a search up inheritance chain
  - Virtual calls cost more than final/static calls
- HotSpot 1.3.1 is conservative
  - Simple chain of inheritance
  - Assumes that non-final methods will be overshadowed

CHA—Class Hierarchy Analysis

- Dynamic determination of inheritance
  - Faster type check (instanceof)
    - Leaf classes are the same as final classes
    - Simple compare for final classes
    - Inheritance table allows indexing
    - Each class has an inheritance depth index
  - Method level analysis
    - Direct calls instead of virtual/interface calls
    - Inlining across virtual/interface calls

Dynamic Compilation

- Scenario
  - Better code generation by making assumptions using current state
  - Dynamic loading of classes force change
- HotSpot 1.3.1 is conservative
  - Compiler makes no assumptions
  - Methods still work even if new classes are loaded

Dynamic Compilation

- Compile using current state
- Dynamic replacement
- OSR—replacing interpreted with compiled
- De-optimization—replacing compiled with interpreted
    - Correct for changes in class hierarchy (CHA)
    - Changes in optimization level
    - Debug compiled methods

Full-speed Debugging

- Scenario
  - Debugging interpreted is too slow
    - Example: 20,000,000 data points calculated before bug occurs
- Debugging in 1.3.1 forces interpretation across the board (-Xint)

Full-speed Debugging

- HotSpot compiler not disabled during debugging (-Xdebug)
- Breakpoints and single stepping force de-optimization

Aggressive Inlining

- Scenario
  - Inlining removes cost of method call
  - Inlining provides better cross method optimization
- HotSpot 1.3.1 inlines simple accessors, statics and final methods

Aggressive Inlining

- HotSpot 1.4 inlines
  - Increased complexity and depth
  - Uses CHA information
  - Knowing that we can always de-optimize
  - Significant improvement in performance

Intrinsics

- Core methods
  - Reduce JNI call overhead
  - Better native implemention
  - Performance bottleneck
- HotSpot 1.3.1 few intrinsics
  - sin/cos/sqrt
  - java.lang.String.charAt

Intrinsics

- NIO—accessing external memory directly
  - Direct memory accessors
  - Eliminates per byte accessors
- Per thread allocation is inlined
- instanceof and checkcast utilize CHA
- Fine grained monitors are inlined

Low-level Intermediate Representation (LIR)

- Scenario
  - Need to provide better platform-specific optimization
- HotSpot 1.3.1 compiles directly from high-level intermediate representation
  - Too coarse grained

Low-level Intermediate Representation (LIR)

- Supports
  - Near native instructions
  - Some support for higher level functionality
    - Virtual calls, Type checks, Intrinsics
- Optimizations
  - Peephole optimizations
  - Better register allocation
  - Unsafe array access

How to Utilize Improvements

- Write small concise methods
- Accessors are inlined
- final is superficial as far as optimization concerned
  - Let CHA do the work
- Avoid object pools
  - new is inlined/per thread allocation
  - Fast allocation and precise collection

How to Utilize Improvements

- Avoid programming by exception
  - Exception object creation is expensive
    - Trace back
  - Exception handling inhibits optimization
  - Exception handling costs only when used
- HotSpot Compiler optimized for clean OO code
  - jikes for development, javac for deployment
    - HotSpot is optimized for javac patterns
  - Obfuscation is often costly

New API in Java 2 Standard Edition, v1.4

Victor Hernandez
Technical Lead, Java VM Team

New API in J2SE v1.4

- Introduce new API
- Performance opportunity: New I/O
- Changes to Java Native Interface (JNI)
- Demo

J2SE 1.4: New Non-GUI API

New I/O:
- java.nio
- java.nio.channels
- java.nio.charset

XML:
- javax.xml
- org.w3c.dom
- org.xml.sax
- org.apache.crimson
- org.apache.xalan
- org.apache.xml
- org.apache.xpath

Miscellaneous:
- java.util.logging
- java.util.prefs
- java.util.regex
- javax.security

J2SE 1.4: New Non-GUI API

New I/O:
- java.nio
- java.nio.channels
- java.nio.charset

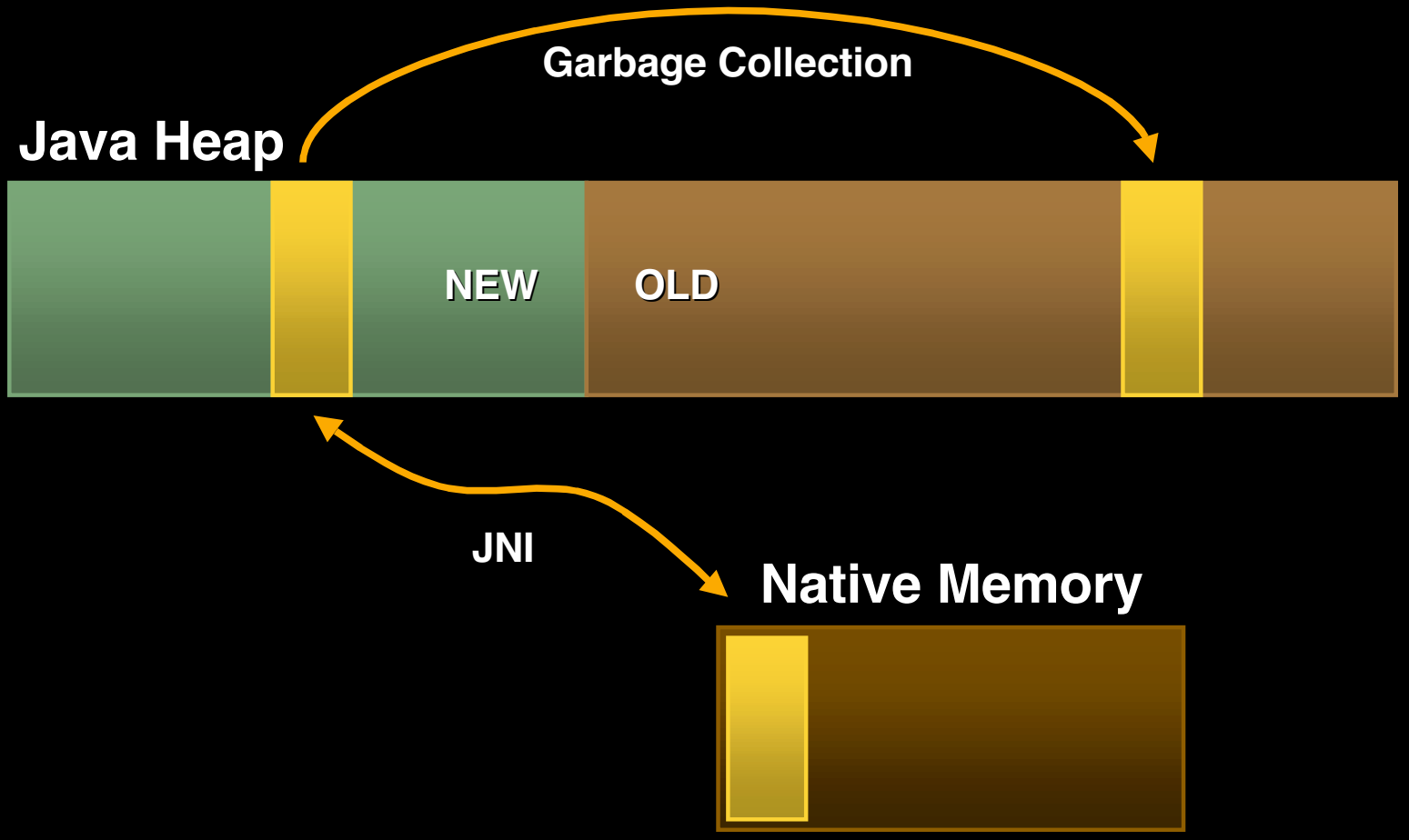
XML:
- javax.xml
- org.w3c.dom
- org.xml.sax
- org.apache.crimson
- org.apache.xalan
- org.apache.xml
- org.apache.xpath

Miscellaneous:
- java.util.logging
- java.util.prefs
- java.util.regex
- javax.security

J2SE v1.4: New I/O

- Problems with present I/O (java.io)
- Problems solved with New I/O (java.nio)
- Using New I/O
    - Buffers
    - Channels
    - Unicode support
    - JNI support for New I/O

Present Native Memory Access



Problems With java.io

- No direct access to native memory
  - Every buffer has a native copy
    - Copying has JNI and CPU overhead
    - GetPrimitiveArrayCritical has GC overhead
- Garbage Collection moves buffers
- Can not memory-map files
- Only supports blocking I/O
- *Does not scale well*

Problems Solved With java.nio

- Allocate native memory from Java
- Direct access to native buffers
  - Integrated into HotSpot JIT Compiler
- No Garbage Collection or JNI overhead for copying
- Enables non-blocking I/O
- *Achieves native I/O performance*
- java.nio supplements, not replaces, java.io
  - But not yet used throughout JRE

J2SE 1.4 New I/O: Native Buffers

- Native Buffers
  - java.nio.*Buffer
  - Containers for data of primitive types
  - Access through get/put methods
- I/O operations on java.nio.ByteBuffer
  - Direct
  - Memory-mapped files
- Allocation/deallocation costs differ from Java arrays

J2SE 1.4 New I/O: Channels

- Channels
  - java.nio.channels.*Channel
  - Allows non-blocking and interruptible operations
- Improved file system support
  - java.nio.channels.FileChannel
    - File locking
    - Memory mapping
- Inter-process communication
  - java.nio.channels.SocketChannel
  - java.nio.channels.PipeChannel

New I/O: Unicode Support

- Character set support
  - java.nio.charset.*
  - Decoders: bytes to Unicode characters
  - Encoders: Unicode characters to bytes
  - Operate on ByteBuffer and CharBuffer

J2SE 1.4 New I/O: JNI Changes

- New JNI functions

```
jobject NewDirectByteBuffer(JNIEnv *env, void
   *address, jlong capacity);
void* GetDirectBufferAddress(JNIEnv *env,
   jobject buf);
jlong GetDirectBufferCapacity(JNIEnv *env,
   jobject buf);
```

- Old JNI libraries are compatible with JNI 1.4

Demo

J2SE 1.4 Native I/O: Pixel Blitting

Victor Hernandez
Java Runtime Team

Summary

- Discussed performance of your Java application
- Improvements in HotSpot JIT Compiler
- Performance opportunities with J2SE 1.4 APIs

Who to Contact

Alan Samuel
Java Technologies Evangelist
bluker1@apple.com

Allen Denison
Java Product Manager
allen@apple.com

Jim Laskey
Technical Lead, Java VM Compilers
jlaskey@apple.com

Victor Hernandez
Technical Lead, Java VM Team
vhernandez@apple.com

http://developer.apple.com/wwdc2002/urls.html

For More Information

- Apple Java
  http://developer.apple.com/java
- Java 2 Standard Edition, version 1.4
  http://java.sun.com/j2se/1.4
- Top Ten Cool New Features of Java2SE 1.4
  http://www.onjava.com/pub/a/onjava/2002/03/06/topten.html

How to Access Documentation

- Most up-to-date: PDF and HTML
  http://developer.apple.com/techpubs/java
- Hardcopy print-on-demand
  Vervante.com under Related Resources
- Product CD
  Documents folder and installed in
  /Developer/Documentation/Java
- Check ADC News for latest updates
  http://developer.apple.com/devnews

Links and Resources

- Apple Java Developer Resources
  - http://developer.apple.com/java
    - New Documentation!
- Java-dev Mailing List
  - java-dev@lists.apple.com
- Developer Technical Support
  - dts@apple.com

Members of Java Runtime Technologies Team

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002